Express Mail No.: EL622260879US Date of Deposit: 08/17/2001

APPLICATION FOR UNITED STATES LETTERS PATENT

FOR

SYSTEM FOR ON-LINE FINANCIAL SERVICES USING DISTRIBUTED OBJECTS

Inventors: William P. Anderson
Jacob B. Geller

Assignee: Block Financial Corporation

Attorneys: Standley & Gilcrest LLP
Attn.: Jeffrey S. Standley
495 Metro Place South
Suite 210
Dublin, Ohio 43017
Phone: (614) 792-5555

SYSTEM FOR ON-LINE FINANCIAL SERVICES USING DISTRIBUTED OBJECTS

Inventors: William P. Anderson
Jacob B. Geller

This application is a continuation of U.S. Patent Application Serial No. 09/520,140, filed March 31, 2000, entitled SYSTEM FOR ON-LINE SERVICES USING DISTRIBUTED OBJECTS, which is a continuation of U.S. Patent Application Serial No. 08/902,239, filed July 29, 1997, entitled SYSTEM FOR ON-LINE SERVICES USING DISTRIBUTED OBJECTS, which is now U.S. Patent 6,131,115, issued October 10, 2000, which is a continuation of U.S. Patent Application Serial No. 08/580,074, filed December 20, 1995, entitled SYSTEM FOR ON-LINE FINANCIAL SERVICES USING DISTRIBUTED OBJECTS, which is now U.S. Patent 5,706,442, issued January 6, 1998.

BACKGROUND AND SUMMARY OF THE INVENTION

The present invention relates generally to client/server computer systems. Particularly, the present invention relates to a client/server architecture for delivering financial services to customers of various financial institutions.

Customers of various types of financial institutions such as banks, stock brokerages, credit card companies, and insurance companies often have a need to access information regarding recent account activity or their account balances. Typically, financial information is reported to customers in the form of monthly statements that list the account's activity and balance for the previous month. By the time these statements are processed and sent, they no longer reflect the current state

of the account. Account balances may change on a daily basis for a variety of reasons including the addition of interest earned or the processing of a new transaction.

Customers in need of more timely information regarding their accounts usually have the option of calling a customer service representative of the financial institution to request a balance or activity report. Although the information is timely, it may be difficult or inconvenient to obtain. First, customers must call each institution from which they would like to obtain current information. When calling, they may need to wait for someone who can help. At other times, they may be required to traverse many levels of an automated attendant before reaching an option that will allow them to accomplish a specific task such as obtaining a current account balance. In either case, the information is presented verbally rather than in a written form that more closely resembles a statement. Finally, whether the information is communicated verbally or through a written statement, customers who wish to use the information in a computer program must enter it manually. In addition to the inconvenience, the process of manually entering the data is also error prone.

Customers of various financial institutions therefore, have a need to access recent financial information at their own convenience—preferably, from anywhere and at any time. Furthermore, customers have a need to see the financial data presented in an organized and understandable format similar to the monthly statement format with which customers are familiar. The present invention—Conductor℠ System Architecture (Conductor)—supports a suite of on-line financial services from various financial services providers. Supported services include credit card account lookup and reporting, and checking and bill paying. In addition, customers and financial services

providers may communicate with each other. Finally, the financial information obtained electronically may be downloaded directly to customers' personal computers for further processing. The need for manual data entry is eliminated.

The present invention is a sophisticated computer software system based on distributed system technology. Within the system, use of the TCP/IP protocol suite for communications with major components of the system allows the financial services to be accessed through the Internet. The same services may also be accessed directly through an on-line information service such as CompuServe®. Conductor supports a distributed "information cluster" located on the global Internet so it may be accessed at any time from around the world using any one of a number of presentation tools. A variety of financial services from a number of independent financial services providers are supported by the system so that users may review activity and balances relating to different types of accounts. The ability to use a variety of presentation tools to access a suite of financial services supported by a variety of financial services providers is unique to the present invention. The advantages of the present invention and others are explained further by the accompanying drawings and detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a diagram of the Conductor Network illustrating the components of a financial information service system based on the Conductor System Architecture;

Figure 2 is a block diagram of the Conductor System Architecture; and

Figure 3 is a flowchart of the primary steps of the present invention.

DETAIL DESCRIPTION OF PREFERRED EMBODIMENT(S)

The Conductor System Architecture (Conductor) and its related protocols provide a robust suite of on-line Interfaces for use by applications, financial service providers, Web (hyper-text transfer protocol—HTTP) servers, and other clients to obtain and manipulate financial information for users of the system. Applying principles of modularity and abstraction, distributed systems technologies are used to define the major components of Conductor and their interrelationships to allow delivery of diverse types of financial services over a wide area network. Sources of data may be as varied as the Interfaces to it. Financial information systems using the approach of Conductor are easily extensible because Conductor is based on a platform-portable, language-independent distributed object framework. Client components and server components work in concert to provide timely financial information to users of an on-line financial information system built using Conductor. Use of the distributed approach of a client/server model permits the easy integration of new services and providers for the system. For example, server components of Conductor may easily serve as back-end resources for existing on-line service providers. The distributed approach also allows applications running in the system to be accessible through a number of presentation tools or users interfaces (collectively, clients): for example, native Microsoft® Windows® applications, Web (hyper-text mark-up language—HTML) browsers, text-terminals, X.25 transactions, even voice telephony.

Referring to Figure 1, a diagrammatic representation of the Conductor Network is shown. The Conductor Network illustrates use of the Conductor System Architecture to provide a suite of financial services accessible through different user interfaces.

Preferably, users connect to the suite of on-line financial services in the Conductor Network via the Internet 12. Methods for providing services via the Internet are well-known in the art and are not explained here. Host computers in the network are accessible world-wide from any site with TCP/IP name resolution and packet routing to the *conductor.com* domain. Preferably, host computers running the Windows NT™ Operating System and the UNIX® Operating System are used in the distributed environment. Clients and servers may run on any of twenty operating systems. Multiple user interfaces to applications that are part of the Conductor Network are implemented as different types of clients. As shown in Figure 1, a user may communicate with a financial application via a Web (hyper-text markup language-HTML) browser 10 or via the CompuServe Information Service 14 using the CompuServe Information Manager for Windows® (WinCIM®) 16. Other methods of access may be used as well—for example, a native Microsoft® Windows® application. In addition, Conductor components may include financial services that are part of an on-line information service so that they are available only to subscribers of the on-line information service.

As shown in Figure 1, packets destined for the Conductor Network are routed 18 to a Web Server 22 for processing. Because security is a significant issue for on-line financial information systems, a Firewall 20 is established between the Router 18 and the Web Server 22. User verification and data access may then occur in a secure environment. Separate user connect/data access protocols isolate internal/external networks. An indirect method of user identification is used to secure account numbers

and sensitive data are passed via two-key encryption. Token passing is used for connected host identification.

The Conductor System Architecture is itself built on a Common Object Request Broker Architecture (CORBA)-compliant Distributed Object Computing Platform. This development platform is well-known in the art and is not explained here. Primary system components include Financial Object Servers, Distributed Name (or Name Lookup) Servers, and Database Servers. Other components include Communication, Security, and Logging servers. As shown in Figure 1, a number of Distributed Name (or Name Lookup) Servers 24, 26, 28 and Financial Object Servers 30, 32, 34 may be in operation at one time. When running, these servers may communicate with a Legacy System 38 or other Database Servers 36 in order to respond to specific requests for information. Data requests may be serviced in any one of a number of ways. For example, data may be accessed using a Microsoft® SQL Server running on Windows NT™.

Clients and servers in a Conductor based system communicate according to an application-level protocol. The application-level protocol specifies how a client interprets data sent to it by a server. Differences in the implementation of various services are hidden behind this consistent API. Within applications, the protocol for communication between various components is a call-level API. When one part of the application needs something, it calls a procedural interface in another part. Such calls do not return until the procedure has executed so the flow of control is simple and direct. Extending these synchronous procedure calls across the network interface has

the advantage of simplifying the access to distributed resources by elevating it to the level of standard procedural mechanisms familiar to a majority of developers.

Clients in a Conductor system have an object-oriented Application Programming Interface (API) to the distributed resources or services using a class-like construct called an "Interface" which groups operations and attributes. Interfaces are used by applications, financial service providers, Web (hyper-text transfer protocol-HTTP) servers, and clients to obtain and manipulate financial information for users of the system. Because clients know only the nature of the Interface, it may be implemented in any manner. For example, Interfaces may be implemented in one language and clients in another. The implementation of an Interface may then be altered at will without affecting any clients. As long as the protocol to the Interface is stable, the client implementation is stable.

Clients located anywhere on the global Internet ask for and bind to services by name. Clients locate Interfaces by naming a server which implements one, and they may do so from any site with a TCP connection to the Conductor domain (*conductor.com*). The names of servers are provided by a name lookup Interface which runs on the only host whose name clients need to know. Following name lookup, a client begins communication with a server capable of servicing the client's specific request. The access is synchronous and call-level using either C++, Smalltalk, or C. In other words, clients access services by making standard synchronous procedure calls. Client load is automatically apportioned among all ready object servers at lookup time.

There are several benefits to using name lookup to connect clients and servers. A name lookup layer isolates clients from the location or readiness of any individual

server. Although the financial information system is based on the Internet Protocol (IP), clients are completely isolated from back-end data sourcing concerns and do not need to know the IP addresses of servers. Using this approach, servers may be added simply by connecting to the network, installing system and server software, and adding the machine name to the lookup database. Consequently, clients are not affected by database, network, operating system, hardware platform, or server architectural changes. For example, native 32-bit Windows® applications may use client-side abstraction libraries that hide details of binding to and executing calls on remote servers. Servers may be implemented on cheap, fast Intel-based Windows NT™ network servers and new servers may be added to the system by copying files over and adding the host name to a single locator file. The distributed nature of the system means that it is composed of relatively simple applications that implement a single Interface or a small group of Interfaces through which clients and servers communicate.

Another benefit of using name lookup to connect clients and servers is that servers may have geographical independence. Site independence for servers means that different servers may be developed and maintained by different financial services providers. User access mechanisms provided by clients remain the same so users may access new financial services using familiar methods.

The interface between clients and servers is binary. For various reasons, a binary interface to information and services is preferable to a textual one. Such an interface is more efficient and the data may be useful in more varied applications. Binary data may be converted to text for viewing by humans, sent in binary form to other providers, or retrieved in binary form and processed by a consumer application.

Binary objects may be dragged off of a window and dropped into a finance application or they may be used to generate reports.

Referring to Figure 2, a diagram of the client and server components of a financial information system based on the Conductor System Architecture is shown. Among the server components supported by Conductor are databases. For example, financial information of interest to users of the system is contained in different databases 52, 58, 40 within the distributed environment. Each database has its own access mechanism 50, 56, 62. As explained earlier, among the methods for accessing a system based on the architecture are a Web (hyper-text markup language-HTML) browser 10 that communicates through a Web Server 22 or a native Windows® application 14.

Regardless of the user interface or client in operation (e.g., Web browser 10 or Windows application 14), a financial information request that includes the name of a financial information service 42, 44 may be transmitted from the client 10, 14 to be processed by the name server 24. In the case of the Windows application 14, the financial information request 42 may be transmitted directly to the name server 24. In the case of the Web browser 10, the financial information request may be processed through a Web server 22 that communicates with the name server 24 to determine the location of the financial server to process the request. This approach therefore allows financial services to be implemented as objects and distributed throughout a wide area network such that they may be found through the name server 24.

The Firewall 20 increases system security of applications running in the Conductor environment. The TCP/IP protocol stack 46 is the Internet communication

vehicle. Another Conductor component—the Object Request Broker (ORB)—is an "information bus" that connects clients to the servers or *objects* they need in a heterogeneous environment. By definition, an ORB is platform independent, language neutral, and may run in many networked environments. In other words, ORBs provide interoperability between applications on different machines in a heterogeneous environment. ORBs implemented in one language may communicate with those implemented in another, on a completely different hardware platform. The same is true for the object implementations to which the ORB provides access. Three example objects are shown in Figure 2—a card object 48, a checking object 54, and a bill pay object 60. The objects serve as links between clients 10, 14 and data contained in the databases 52, 58, 40. The name server 24 performs the name lookup function for clients so they may establish communication with the financial object that performs the needed services.

Referring now to Figure 3, there is shown a flowchart of the primary steps of the present invention. Initially, name-financial server pairs are defined and loaded into a name lookup database on the name server 100. Name-server pairs may be added and/or modified as needed. Next, a user may be prompted for financial information such as the user's primary bank, account numbers, type of information desired (e.g., recent debit card transactions), etc. 102. The client with which the user is interacting (e.g., Web browser) may then create an electronic financial information request comprising the financial information provided by the user and the name of a financial information server that can provide the requested information 104. The electronic financial information request is then transmitted from the client to the name server 106.

A database look-up is performed in accordance with the name contained in the financial information request 108. If the name of the financial information server is not found in the database 110, an error is reported to the client 112. If the name of the financial information server is found in the database 110, the financial information request is transmitted to the financial information server located during the database look-up 114. The financial information server then processes the financial information request 116 and the processed information is transmitted back to the client and ultimately, the user 118.

The distributed nature of the Conductor System Architecture means that a financial services system may be composed of relatively simple financial services applications accessible from one of several interfaces. The result of this is that each financial service application is easier to develop and maintain, and the Conductor-based financial services system at large is more flexible and robust. The present invention has been described in the form of preferred embodiments. However, several modifications and variations may be made to the invention and fall within the scope of the claims.